MODIS SEMIANNUAL REPORT
- JULY 1995 -

UNIVERSITY OF MIAMI
RSMAS/MPO

DR. ROBERT H. EVANS

NAS5-31362

===

Due to the interlocking nature of a number of projects, this and subsequent reports will contain coding to reflect the funding source. Modis funded activities are designated with an **M**, SeaWIFS with an **S**, and Pathfinder with a **P**. There are several major sections within this report; Database, client/server, matchup database, and DSP support.

A. NEAR TERM OBJECTIVES

A.1 Modis Objectives (M)

A.1.1. Continue to develop and expand the processing environment

a. increase computational efficiency through concurrent operations

b. determine and apply more efficient methods of data availability for processes

A.1.2. Begin extensive testing using global CZCS and AVHRR GAC data with database processing to test the following:

a. algorithm capability

b. machine and operating system stability

c. functionality required for the processing and analysis environment

A.2 SeaWIFS Objectives (S)

A.2.1. Continue testing of processing methodology.

A.2.2. Continue to develop relationship between database and *in-situ* environment.

A.2.3. Continue to expand the *in-situ* database.

A.3 Pathfinder Objectives (P)

A.3.1. Expand matchup database as applicable.

A.3.2. Continue testing of methodology.

A.3.3 Train and integrate new personnel into Matchup Database processing scheme.

A.4 DSP Objectives (M)

A.4.1. Continue testing of processing methodology.

A.4.2. Continue to expand the number of sites supported.

A.4.3. Expand the supported hardware/software platforms

B. OVERVIEW OF CURRENT PROGRESS

B.1 Automatic Processing Database (P)

B.1.1 Operational Testing

B1.1.1 January Operational Testing

January Operational Processing

Data for 1991 is now available on our new digital tape library. A large number of new pieces of equipment have been added to the processing environment, and the processing slowed while these were integrated into the processing.

A new 4-processor DEC 2100 computer was tested as the primary processor, but numerous problems were encountered. Near the end of the month, the processing was returned to the single-processor alphas.

B.1.1.2 February Operational Testing

Processing continued on the 1991 time period, but was slow due to sever disk and interface problems.

B.1.1.3 March Operational Testing

Processing continued in the 91 time range, covering 91160-91250, using the yearly coefficients for the atmospheric correction. This covered, for the first time, the post-Pinetubo time period. This processing stream was terminated when it became clear that the yearly coefficients were not producing acceptable results for this period.

Processing was discontinued on Modis, the DEC 2100, when it became clear that disk controller problems were severe, and would not allow continuous use during the processing. Processing was returned to four single-processor Alphas.

Two more single-processor alphas were added to the stable of machines processing orbits, totaling six. Subsequently, the pace of the processing increased.

The time period 91001-91166 was recalculated, using the new technique of monthly coefficients for the atmospheric correction step.

B.1.1.4 April Operational Testing

The new, monthly coefficient method was continued extending last month's 91001-91166 out to 91305. At this point, the coefficients were changed, and the time period 91150-920450 was calculated. After another change, 91001-91200 was again rerun.

B.1.1.5 May Operational Testing

Equipment problems were quite severe until a disk reorganization near the end of the month, which alleviated most of the disruptive effects.

During the month, a new, 3-SST algorithm was implemented (see May Development). Days 91001-91324 were calculated.

B.1.1.6 June Operational Testing

Using the new processing scheme, (implemented early in June), the pf3 files (3-SST estimates per file) continued, calculating days 91325 through 93222, or nearly two years for the month.

B.1.2 Development

B.1.2.1 January Development.

There were numerous changes, both small and large, to accommodate the new equipment, which included a 4-processor Sable computer, a digital tape library (used both for spooling input data and backup of output files), and a large amount of new disks.

However, the increase in capabilities also caused a commensurate increase in system problems. In particular, the problems previously encountered with "drop out" of NFS-mounted disks became untenable when Modis was used for processing. The most damaging point was the *.sh and *.dsp files that are written by the VMS APServer and read by the UNIX processor. While there had previously been occasional dropout at this point., this step was failing 3-5% of the time. A new technique was implemented, whereby the APServer writes the *.sh/*.dsp files to a local disk, rshells a job to the processing computer, which transfers the command files to one of its local disks. This method did overcome the problems at this step.

Since the input data files will now be supplied by the DLT device (digital tape library) on UNIX, a number of new procedures and command files were developed to spool the data off, and copy it to the VMS side.

As we develop experience with the new paradigm, more changes will be needed. For example, modifications will be needed to eliminate the need to copy the input file to the VMS disks. Currently, the GETSCAN program extracts the scan lines of the pole crossings, and stores this information in a flat file containing information for a whole year for a given satellite. This scan line information is then extracted from the flat file by the ADDREC program, which decides the pieces to be processed, using the asc/dsc pole crossings, and a set of rules for piece size. This information is used to add the processing records to the database.

We are developing a version of GETSCAN that combines elements of both the current GETSCAN and ADDREC. It uses the pole crossing information to find the scan lines of the pole crossings, breaks the pass up into a few pieces, and writes an ingester input file for each piece.

To fuse the two systems, changes will need to be made to GETSCAN and to ADDREC. Much of the functionality, exclusive of the actual record addition, must be moved to GETSCAN, to eliminate the need for a copy of the input file. These include

> Extracting the exact ingester name corresponding to a scan line.
> Incorporating the full set of rules to define orbit segments.
> Extract any other information from the datafile that is needed.

While the list is small, the information is not so easily defined. We will need to examine ADDREC to select only those programs sections that are currently needed.

Currently, the presence of the input file starts the GETSCAN/ADDREC process. We will have the new GETSCAN put out one ASCII file that contains the information needed for record addition. This file will be rcp'd to the VMS input directory where its appearance will start the ADDREC process. This file will be used by a new _VERY_ streamlined version of ADDREC to add the processing records to the database.

Summary:

	Current	Development	Fused
read asc/dsc file	`GETSCAN	GETSCAN	GETSCAN
find scan lines of pole crossings	`GETSCAN	GETSCAN	GETSCAN
find filename of that scan line	`GETSCAN	GETSCAN	
store scn ln/flnm of pole crossing	GETSCAN		
read scn ln/flnm pole crossing info		ADDREC	

decide on pieces to be created	ADDREC	GETSCAN	GETSCAN
store information on pieces	GETSCAN	GETSCAN	
read information on pieces		ADDREC	
add records to db		ADDREC	

These changes will be implemented in the next few months.

B.1.2.2 February Development

Most development work in this time period still related to the integration of the new equipment, and modifying the processing methods to both make best advantage of the new capabilities and to overcome problems.

Processing this month continued on the 4-processor Sable, Modis, but many more adjustments were needed.

File transfers using rcp were becoming unreliable, therefore the code and command files will need to be changed to use ftp for file transfer.

A major revision of the command files and procedures is in progress. These will be covered in the next month's report, when they are complete.

B.1.2.3 March Development.

All file transfer was changed from rcp call to ftp.

There has been considerable consolidation and shortening of commands.

Many of the commands can be issued from any computer - i.e., even checking and control of the server status can be done from UNIX.

There used to be a large number of single-use command files - many of these (but not all) have been consolidated into single command files that interpret input parameters. These changes will be covered in a separate section.

The new system will not seem much different. The major differences OPERATIONALLY are changes in the daily and weekly job triggering, changes in some directories, and changes in how mcp is started.

MCP:

The old need for different mcp binaries for each job type has (finally) been eliminated. A single mcp is used, and the 'psa' command has been modified to show the command line in the ps grid, showing which job type is being run.

The mcp-t job, which ran the recipe called GAC_PTB, has been changed to reflect its function - it is now mcp/orbit, and the recipe name is GAC_ORBIT.

Previously, there were separate files to start each job or combination of jobs (starti, startu1, startust, etc.). There is now one 'start' command file that takes a single parameter, the type of jog or jobs to start. Currently, the choices are:

Single: i, u1, s1, o, d, w

Combined: uso - u1, s1 & o
us - u1 & s1

The start command file can be rsh'ed to the machine you want to start from andrew. I do not have comparable versions of the files like startenu, startkel, startall and loadenu1, etc.

DIRECTORIES

The pass_time and asc/dsc files are now in a directory pointed to by ap_etc, instead of mcp_etc. The ap_autoproc_computer_##.sh .dsp and .rpt files, as well as the data transfer files for the record adder now go into dsp_usr2:[ap.tmp.alexis]. (Or in dsp_usr2:[ap.tmp.mariah] for the mariah system).

DAILY/WEEKLY TRIGGERING

Previously, when all passes for a given yearday were done, the daily job for two days before was triggered. This permitted daily jobs to run before all passes for that day were processed, causing stray orbits to pop up at times.

The new triggering is:

When each orbit is done, the server checks to see if that day is complete. If not, no triggering occurs. If so, then:

Three day periods are checked to see if a daily job is ready. For example, say day n has just been completed:

Check if day n-1 is done. If so, check if day n-2 is done. If n, n-1 and n-2 are done, trigger daily for n-1.

Check if day n+1 is done. If so, check if day n+2 is done. If n, n+1 and n+2 are done, trigger daily for n+1.

Days n-1 and n+1 were checked. If n-1, n and n+1 are all done, trigger daily for day n.

Weekly:

When a daily job completes, the server checks to see if all days of that week are done. If so, the weekly1 job is triggered.

When the weekly1 job is done, the server checks to see if the previous and next weeks are done. If so, the weekly2 job is triggered.

Completion of the weekly2 job triggers the weekly3 job, which is allowed to run only on the alpha servicing the vol set for that week.

Completion of the weekly3 job triggers the weekly4 job.

VMS Commands

The old way to start the VMS batch jobs has been shortened a bit. The old '@ap_com:submit_apserver' command has been replaced by a simple 'apserver'.

All the new commands to start UNIX are:

```
$ apserver(was: @ap_com:submit_apserver)
$ addrecgac(was: @ap_com:submit_addrecgac)
$ getscangac(was: @ap_com:submit_getscangac)
$ fixscangac(was: @ap_com:submit_fixscangac)
$ mvgetscan(was: @ap_com:submit_mvgetscan)
```

Also as before, the MCP control is in the dbrequest_* and run_* logicals, with the DBREQUEST_STATUS controlling the mcps from all processing machines., and the RUN_abrv controlling the mcps from a single machine (i.e., RUN_KEL controls KELSO).

However, now a single command file can manipulate both the dbrequest_* and run_* logicals. It is 'mcp.com'. If you type only 'mcp', the current run parameters are listed. If you type 'mcp ?' a short explanation is printed. To change the global value, input the change as the first input parameter. (To stop mcp's , type 'mcp stop'. To change the values for only one computer, use 'mcp option computer'. You can use either the full computer name or an "accepted" abbreviation. (The "accepted abbreviations are listed in the db loading file ap_dbdef:computer_load.sql. This file can be found AFTER invoking the home:ap_build.com command file, which defines the directories for the db and server.)

For checking the db records, there are still a number of *.sql files that can be used inside SQL, or are called by a DCL command file. However, they are all called by a single command file, ck.com, which also has a symbol defined, therefore from ANY directory, you simple invoke ck and input a parameter. This ck command file first checks to see if the parameter is one of its "known" commands. If it is, ck calls the right command file in SQL. If the input parameter is not recognized as a command, it is assumed to be the name or abbreviation of a computer, and the jobs assigned to that computer are checked. The queries that are recognized are:

```
MAINLINK (can input a main_link, or be prompted)
DAILY
DSPA
EXE
INIT
ORBIT
```

ORBIN (BOTH ORBIT AND INIT)
SUB
HSUB
UNF
WEEKLY
WEEKLY1 or W1
WEEKLY2 or W2
WEEKLY3 or W3
WEEKLY4 or W4
FOURWK or FW
WEEKLYSUB

Note that the MAINLINK query can take a main_link as the second input parameter, but that ck prompts for it if omitted from the command line.

VMS/UNIX CROSSOVER

Both mcp and ck can be issued from any UNIX machine, as well as on the VMS server. The "mcp" command is just passed directly to the server machine. However, there are a number of "ck"-type commands that are also performed on the UNIX side, and these are tried first, then the ck command is passed to the VMS server if the UNIX command file doesn't recognize the input parameter.

UNIX Commands

This leads to the UNIX command files. First, they no longer reside in the ap directory. they are now in a /usr/dsp/com2 directory (just as the executables are in /usr/dsp/bin2).

The pathgac* , *ingest* and rcp_control.sh files are used by the processing, not interactively. The ap.defs files just sets up the environmental variables for the processing. That leaves just three files:

exa - runs the full examin on an image (to retrieved full header)

start - starts the processing on various machines

ck - performs various kinds of checks

The start command file should take two parameters: the type or types of mcps to start, and the computer to start them on. The remote start is not quite working. The type(s) of mcps to start are:

```
u1     s1     o     i     d
w1     w2     w3     w4     fw     ew
u2     u3     u4     s2
us     uso     ii     iii
3uso    4uso    4u2so   usoii   usoiii
```

the ck command file checks for a number of options to check on the UNIX side. If the command is not recognized as one it knows, it passes the whole command line to the VMS side.

Processing Overview

The input dataset consists of a stream of datafiles representing (roughly) orbits of a satellite. As the satellite moves forward, its sensor scans left and right, recording two scans per second (for GAC) The satellite orbits the earth in about 100 minutes. Each orbit is recorded on tape recorders that provide an approximate 5 minute overlap between the end of one recording and the initiation of the next. This overlap needs to be identified and eliminated from the resulting products.

After about 14 orbits, the earth has rotated once (i.e., one day has passed). This means that the satellite has taken data over the entire earth TWICE, once on an ascending (south to north) leg and once on a descending (north to south) leg. For the NOAA polar orbiters, the ascending leg corresponds to daytime and the descending to night.

On the archive device, these data files are stored in increments of a yearday (the first five digits of the filename), and the various jobs that store, retrieve and access the data are usually organized in the manner. To date, this has been the most convenient arrangement, with one day's worth of data fitting comfortably on the staging disks while still "occupying" the processing streams.

The processing scheme can be though of as addressing a number of tasks, which can be sorted into characteristic time and/or space scales, and the tasks or tasks that must occur at each of these scales, and the order in which they must occur. Each level has a dominant scale, but that scale may be either temporal or spatial or both.

In the AVHRR GAC processing, the tasks and their dominant scales are

Task Increment Components

1. Spooling data: yearday 14 orbits
2. Autoproc entry: orbit self-contained
3. Ingest/atmospheric corr: orbit->6 pieces @ 2400 scan lines
4. Spacebin: each orbit segment
5. Orbit: construct from orbit segments
6. Daily: 3 yeardays14 +2 orbits for data day
7. Weekly1: 1 week <- 7 daily files
8. Weekly2: 3 weeks <- 3 weekly files
9. Weekly3: 1 week <- weekly reference file
10. Weekly4: 1 week <- cloud-masked files

1. Spooling data: This job acts as a daemon, checking to see if files are needed, and that disk space is available. If so, it copies one day's work of files from one device to another. Most of the spooling jobs are submitted once and recycle in increments of one day.

2. Autoproc entry: This job notifies the db that a data file exists and is ready for processing. It deals with a single file at a time.

3,4,5. Ingest/atcor and spacebin: These jobs work on individual pieces from a single orbit file. They can be thought of as a set of tasks that must be performed in sequence to a single piece. We need to separate the tasks so that parallelism occurs, that is, one piece is being ingested while the last piece that was ingested is being atm. corrected, and the piece just corrected is being spacebinned. (See short section later for explanation of each of these steps.)

6. Orbit: When all pieces of an orbit file have been finished (through spacebinning) a job is run that gathers the pieces into single files for transfer to a remote disk. This consolidation produces (usually) one ascending and one descending file for the given orbit, plus POSSIBLY a third file with data to be included in either the previous or next day. (Orbit segments that cross the 180° meridian will produce results for adjacent “data-days”).

7. Weekly1: When all seven days of a give week have been created, the ascending and descending data are combined into weekly asc and dsc files.

8. Weekly2: When the weekly files for three weeks in succession have been created, these are combined into 3-week asc and dsc files, and these files are run through a gap-filler. These are called the asc and dsc reference files for the middle week of that three-week period.

9. Weekly3 & 4: After the reference files for a given week have been produced, these are used to cloud-mask each daily file for that week (asc ref used on asc dailys, dsc on dsc dailys). The cloud-masked daily files are then recombined into cloud-masked weekly files, and a large number of product files are extracted from the daily and weekly declouded files.

Ingest/AtCor/SpaceBin: Ingest refers to the process of extracting a part of an input orbit and storing it in a dsp-format file, including the navigation format the orbit file itself. After ingest, a series of procedures is run on the ingested file to apply various corrections to the navigation (this is called the sector process, which is actually a series of processes). No new file is produced in the sector process. The atmospheric correction part takes the initial data (radiance in five different frequency bands) and produces an SST estimate at each point in the input ingest file. These files are still at the 4 km resolution, and are in what is called "satellite perspective", i.e., the successive scan lines. The spacebin job takes the input satellite perspective data and bins it into a global 9km equal area grid. Note that these files contain entries only for bins with data. Successive jobs combine or treat the data using these bins.

Database description

There are three sets of tables in the processing database. One set contains three tables that store information on the processing, defining the steps and what is to be done in each of the steps. Another set has two tables that track each of the processing entities and the jobs associated with them, and a third set contains various tables that provide ancillary information for definition and validation purposes.

These types contain these tables:

PROCESSING RECIPE TABLES:

RECIPES
PROCESS_STEPS
PARAMETERS

JOB ENTRY TABLES:

MAIN
PROCESS_CONTROL

MISC. DB TABLES

BOOK_KEEP
COMPUTERS
DAYINFO
SATELLITES
SATSEN
SENSORS
USER_GROUP
WEEKINFO

~

PROCESSING RECIPE TABLES -
RECIPES/PROCESS_STEPS/PARAMETERS:

There are three tables that are used to define the processing stream: the RECIPES table, the PROCESS_STEPS table and the PARAMETERS table. (Future implementations will combine these into either one or two tables.) The RECIPES table contains an entry for each integral step in the processing stream. That is to say, a RECIPE defines a step or set of steps that are to be performed as a unit. The PROCESS_STEPS table contains a set of entries for each RECIPE, defining the processing steps, and the order in which they are to be performed. The PARAMETERS table provides additional definitions that may be used in the PROCESS_STEPS table.

RECIPES Table:

A processing RECIPE may contain more than one processing step. The steps defined by a RECIPE are to be performed as a unit, without interruption, and the results of the processing transmitted to the processing db at the end. The fields of the RECIPE table are used to assign characteristics and control some aspects of the job triggering. The fields are:

Table RECIPES
recipe_code: Code number for keyword retrieval
recipe: Recipe name
def_priority: Default priority to assign to procedure
process_class: All jobs with the same "svctype"
computer_class: When one job is assigned to a computer, all are
trigger_class: Trigger_class of this recipe
class_to_trigger: Trigger_class to trigger when no more sub or fin records

Values for these fields are:

CD	RECIPE	PRI	PROCESS_CLS	COMPUTER_CLS	TRIGGER_CLS	CLS_TO_TRIG
1	'GAC_INIT'	1	'INIT'	'INGATSB'	'INIT'	'INGATCO
2	'GAC_INGATCOR'	1	'UNLIMITED'	'INGATSB'	'INGATCOR'	'NONE'
3	'GAC_SPACEBIN'	1	'SBIN'	'INGATSB'	'SINGLEREC'	'TIMEBIN
4	'GAC_ORBIT'	1	'ORBIT'	'INGATSB'	'TIMEBIN'	'DAILY'
5	'GAC_DAILY'	1	'DAILY'	'WEEKLY'	'DAILY'	'NONE'
6	'GAC_WEEKLY1'	1	'WEEKLY1'	'WEEKLY'	'WEEKLY1'	'WEEKLY2

The process_class field refers to the job type. That is, when mcp requests a job, it requests a job with a particular process_class. This was originally intended to be used if one mcp would be allowed to run different types of jobs, but has operationally become limited to a single type of job for each mcp. In the example, there are six different process_classes representing five mcp types: mcp-i, mcp-u1, mcp-s1, mcp-o, mcp-d and mcp-w1.

The computer_class is used to control which computer performs the processing. For example, the processing of a single orbit should all occur on the same computer, as the input file must be copied to a local disk, and this should be done only once. When an orbit is assigned to a computer, all tasks with the "INGATSB" computer class will only be run on that computer. In the example, when the INIT job is assigned to a particular computer, all other jobs with the same computer_class (INGATCOR, SPACEBIN and ORBIT) are also assigned to that computer.

The trigger_class and class_to_trigger fields are used in the triggering of follow-on jobs. When a job completes successfully, the process_status for that collection of jobs (i.e., all process_control records with the main_link) is checked. If there are no submitted or aborted jobs, then the "class_to_trigger" field is checked for the recipe just completed. If it is "NONE" no triggering of this type is performed; otherwise, all records with the specified trigger_class are submitted for processing. In the example, when the INIT job completes, all INGATCOR jobs for that orbit will be submitted for processing.

PROCESS_STEPS Table

This table defines the information retrieval, workspace definition, and command files to be run, and their order, for each recipe to be used in a processing stream. The fields of the PROCESS_STEPS table are:

- ! RECIPE: Steps are assigned to this procedure
- ! process_step: Process step number
- ! command: Command to execute
- ! jump: Jump to this step if appropriate

Each recipe will have a number of steps defined for it.

RECIPE	PROCESS _STEP	COMMAND	JUMP
'GAC_INIT',	1,	'NAME_MAKE',	0
'GAC_INIT',	2,	'pathgac_init',	0
'GAC_INGATCOR',	1,	'NAME_MAKE',	0
'GAC_INGATCOR',	2,	'GET_WEEKINFO',	0
'GAC_INGATCOR',	3,	'SET_INGEST',	0
'GAC_INGATCOR',	4,	'WSO:nlmc',	0
'GAC_INGATCOR',	5,	'pathgac_ingatcor',	0

There are three type of items that can be defined in a step.

1: Certain commands (such at SET_INGEST or GET_ASCDSC) direct the server to retrieve information from the database, and write out workspace variables. (Examples: Steps 1 in GAC_INIT and 1-3 in GAC_INGATCOR above.)

2: A command may be simply the name of a dsp command file to execute. (Examples: Steps 2 in GAC_INIT and 5 in GAC_INGATCOR above.)

3: A step can be used in conjunction with the PARAMETERS TABLE to assign a particular value to a workspace variable. (Example: Step 4 in GAC_INGATCOR above - see also PARAMETERS below.)

PARAMETERS Table

The PARAMETERS table contains three fields:

command	command for which variable is defined
work_space	workspace variable to define
data_value	value to assign to variable

and sample values are:

COMMAND	WORK_SPACE	DATA_VALUE
'WSO:nlc'	‘SSTTYPE'	'nlc'
'WSO:nlsst'	'SSTTYPE'	'nlsst'
'WSO:nlmc'	'SSTTYPE'	'nlmc'

In the PROCESS_STEPS example, step 4 in GAC_INGATCOR is requesting that a workspace variable, 'SSTTYPE', be assigned a value of 'nlmc' before the dsp command procedure 'pathgac_ingatcor.dsp' is run.

JOB ENTRY TABLES:

MAIN Table

This table tracks the "entities" that require processing. There is one entry for each "class" of jobs that need to be run. for example, there is one record for each satellite orbit to be processed, and the file_name is the input orbit file name. There is another entry for all jobs that need to be run on a daily basis (i.e., jobs that produce files in increments of one day), and another for weekly jobs, etc. The file names of these are yyddd, yyww, etc.,

There will be at least one record in the PROCESS_CONTROL table for each record in the MAIN table. If multiple jobs are to be run on the same "entity", there will be one PROCESS_CONTROL record for each job.

Field	Comment
record:	Main record number
file_name:	Link to several relations
:	File number on archive medium
raw_data_link:	Link to RAW_DATA table
archive:	Validate with ARCHIVE table
archive_label:	Link to ARCH_INFO table
format:	Validate with FORMAT table
satsen_code:	sat/sen code number
satellite:	Type of satellite - validate with SATELLITE table
sensor:	Type of sensor - validate with SENSOR table
transmission:	Transmission - validate with TRANSMISSION table
channel:	Channels in input file - validate with CHANNEL table
orbit:	Satellite orbit number
pass_time:	Timestamp at beginning of pass
pass_end:	Time of last scan line in the pass
yearday:	Yearday of file
scans:	Number of scan lines in scene
miss_scan:	Number of missing scan lines in scene
source:	Data source - validate with SOURCE table
project:	Project - validate with PROJECT table
release_link:	Main record to notify upon completion
release_recs:	No. of release_recs this is waiting for
release_done:	No. of release_recs done
process_recs:	No. of process_control recs for this entity
process_done:	No. of process_control records completed
map:	Bitmap of 10 X 10 degree coverage
QC_status:	Status of quality control
who_code:	Who modified; Validate with USER_GROUP table
last_mod:	Date of last record modification
audit:	Date of record creation

PROCESS_CONTROL Table

This table stores information on individual jobs through the automatic processing system.

FieldComment	
-------------	-------------------------------------
record:	Internal record number of file to process
main_link:	Link to MAIN relation
recipe:	RECIPE to be executed, Validate with RECIPE table
process_step:	Last process_step completed (obsolete)
satsen_code:	
satellite:	Validate with SATELLITE table
sensor:	Validate with SENSOR table
computer:	Computer assigned to process this PCR; Validate with COMPUTER table
computer_class	Used to restrict jobs to one computer
process_status:	Is the hob is in HOLD, SUB, FIN or DSPA status
process_class:	MCP-type class of this job
rec_to_trigger:	PC Record to trigger when this completes if zero, this is triggered by another if -1, neither triggers nor is triggered
trigger_class:	The trigger_class of this job
class_to_trigger	The class to be triggered when this class has completed
priority:	Priority to assign to job
totscan:	Total scans in WHOLE SCENE
begscan:	Beginning scan line of this piece
endscan:	Ending scan line of this piece
yearday:	Yearday from MAIN_LINK's input file
miscinfo:	Miscellaneous information (varies)
source_file:	Output file for this PCR.
source_directory	Directory of input file.
who_code:	Validate with USER_GROUP table
last_mod:	last record modification
audit:	record creation

MISC. DB TABLES

BOOK_KEEP Table

The BOOK_KEEP table is a special-purpose table used to keep track of the current number of entries in those tables that are permitted to "grow. The three fields in the BOOK_KEEP table are:

record	record number in this table
relation_name	name of relation
max_record	current number of records

and INITIAL entries are:

RECORD	RELATION_NAME	MAX_RECORD
1	'BOOK_KEEP'	4
2	'RAW_DATA'	0
3	'MAIN'	0
4	'PROCESS_CONTROL'	0

The BOOK_KEEP entry has an initial value of 4 because four tables are currently used in this manner - the three "data" tables and BOOK_KEEP itself.

When an new entry is added to any of these tables, the BOOK_KEEP value for MAX_RECORD for that table is first incremented, then that value assigned as the record number for that table. This is the APServer's method of assigning distinct record numbers to these tables.

COMPUTERS Table

The COMPUTERS table is used to validate computer names using the APServer system, and to provide information about them. The fields in the COMPUTERS TABLE are:

FieldComment	
computer	Valid keyword value
abbrv	Abbreviation
os	Operating system
site	Location of computer
*The following fields are not currently being used.	
run_status	Running status of computer (run, pause, finish, stop)
alarm_status	Whether the start/stop alarm is to be used (n/y)
alarm_start	When to begin processing (hhmm) (LOCAL time)

alarm_mcps	Which mcps to start (list of start files, sep by comma)
alarm_stop	When to finish up processing (hhmm) (LOCAL time)

and typical entries are:

COMPUTER	ABBRV	OS	SITE
MARIAH	mar	VMS	RSMAS
andrew	and	UNIX	RSMAS
enuka	enu	UNIX	RSMAS

There are also a number of fields that have not yet been implemented, that will be used to automatically start and stop processing on a given machine.

DAYINFO Table

This table is used to define the start the ascending and descending data-days for each satellite. This table also assigns a week to each day (which can be defined separately by satellite). The fields are:

Field	Comment
yearday	yearday
satsen_code	sat/sen code number
week	Week to which this day is assigned
asc_beg	Beginning of day for ascending data
dsc_beg	Beginning of day for descending data

and typical entries are:

```
    SATSEN_
YEARDAY CODE WEEK ASC_BEG     DSC_BEG    YEARDAY CODE WEEK ASC_BEG
DSC_BEG
 88308    1  8845 88308011124 88307135119 88308    2  8845 88308044643
88307...
```

Entries need only be made for a particular satellite for the life of that satellite, not for all days defined in the table.

SATELLITES Table

This table is used to validate satellite names and store associated data. The fields are:

Field	Comment
satellite	satellite name
extension	file extension for ingested files
sat_id	NORAD satellite number
prefix	prefix for processed files

and typical entries are:

```
 SATELLITE    EXTENSION  SAT_ID  PREFIX
'NOAA-9'     ,'NO9',      15427,   'K'
'NOAA-10'    ,'N10',      16969,   'J'
```

The SATELLITES table is linked to the SENSORS table by the SATSEN_CODE table.

SENSORS Table

This table is used to validate satellite sensors.

FieldComment	
sensor	Valid keyword value

```
SENSOR
'AVHRR'   'HIRS2'   'MSU'    'SR'     'VHRR'
'CZCS'    'DCS'     'ALT'    'HCMR'   'MSS'
'OLS'     'SAR'     'SCAT'   'SMMR'   'TM'
'VAS'     'VIR'     'VISSR'  'SEAWIFS'
```

SATSEN Table

This table is used to validate satellite/sensor combinations, and assign a specific code to each pair. It is used to link the SATELLITES and SENSORS
tables.

Field	Comment
satsen_code	sat/sen code number
satellite	satellite name
sensor	sensor name

SATSEN _CODE	SATELLITE	SENSOR
1	'NOAA-11'	'AVHRR'
2	'NOAA-9'	'AVHRR'

USER_GROUP Table

This table is used to validate members of the user USER_GROUP.

Field	Comment
code	Code number for keyword retrieval
user_name	Computer user_name ID
full_name	Users' full name
telephone	Phone
site	Where the user is
email_address	E-mail address
telemail_address	TELEMAIL address

Note: The autoprocessing interface automatically loads the USER_NAME into this table when an new user first runs the system. All other information (full name, site, etc.,) must be loaded separately.

Code	User_Name	Full_Name	Telephone	Site
1.	VICKI	VICKI M. HALLIWELL	305-361-4178	RSMAS

Code	User_Name	email_address	telemail_address
1.	VICKI	vicki@miami.rsmas.miami.edu	NONE

WEEKINFO Table

This table is used to define the 4-week and 8-week time periods, and the directory for the daily files for each week.

Field	Comment
week	The week
fourwk	The 4-week interval for this week
eightwk	The 8-week interval for this week
wkdir	Directory for the daily files by week

```
WEEK FOURWK EIGHTWK WKDIR
  0    104     108  '/(disk)/wk/wk01'
  1    104     108  '/(disk)/wk/wk01'
  2    104     108  '/(disk)/wk/wk02'
  3    104     108  '/(disk)/wk/wk03'
```

...(entries omitted)

The triggering is in the db_report subroutine.

1. When the end of a job is reported to the database, the value of "status" and the existence of an error message is checked. If either of these indicate an error, then the process_status for that process_control record is set to 'dspa', and no triggering occurs. Otherwise, process_status is set to 'fin', and the triggering is tested.

2. If the rec_to_trigger for this process_control record is greater than zero, it is marked 'sub' (submitted for processing).

Example - Completion of Ingatcor triggers Spacebin:

record	recipe	filerec_to_trigger
1001	Ingatcor	90123123456.data1002
1002	Spacebin	90123123456.data-1

The completion of record 1001 will trigger the submission of record 1002.

3. If the rec_to_trigger is less than 1, the "class_to_trigger" for that record is checked. If it is "none", no further class triggering is done. Otherwise, the process_control records for that main record are checked. If any records are marked 'sub' or 'exe, no triggering occurs. Otherwise, all process_control records associated with that main record with the matching trigger_class are submitted for processing.

Example - Completion of Init triggers multiple Ingatcors:

record	recipe	trigger_class	class_to_trigger
1000	Init	Init	Ingatcor
1001	Ingatcor	Ingatcor	None
1003	Ingatcor	Ingatcor	None
1005	Ingatcor	Ingatcor	None
1007	Ingatcor	Ingatcor	None
1009	Ingatcor	Ingatcor	None
1011	Ingatcor	Ingatcor	None

When the Init job complete, all six of the Ingatcor jobs will be submitted.

Three special cases:

If the process_class is "orbit", the orbits for that yearday are checked. If any are unfinished, no triggering occurs. If all are finished, the previous and succeeding days are checked. If all are finished, then the rec_to_trigger of the middle day is submitted. The previous and succeeding three-day periods are also checked if needed.

If the process_class is "daily", all the daily jobs for that week are checked. If any are unfinished, no triggering occurs. If all seven days for that week are done, then the first weekly job (WEEKLY1) is submitted.

If the process_class is "weekly2", the "weekly1" jobs for that week, the previous week and the succeeding week are checked. If any are unfinished, no triggering occurs. If all are finished, then the rec_to_trigger for the middle week is submitted. The previous and succeeding 3-week periods are also checked.

B.1.2.4 April Development

Most of the database processing effort in April was devoted to operational processing.

Much new equipment was installed during this time period, which necessitated minor changes to accommodate the processing. However, in some cases, the equipment continued to cause problems.

B.1.2.5 May Development

After the tests of the monthly coefficients, a test was designed to calculate the SST three ways in a single pass. During the first half of the month development centered on accommodating the programs and command files to handle these calculations. Many minor changes were needed in almost all programs and command files used in the processing, such changes as adding the ability to refer to different SST values (i.e., using "SST" as a default type, and adding "hemsst" and "mnlsst" as additional SST types in a single file).

During the last half of the month, a new method of adding processing records to the database was designed. In the old method, the input file had to be copied to a VMS disk, where information was extracted from it by a batch job on VMS. In the new model, a daemon on the UNIX side will extract the necessary information, which will be transferred to the VMS side in a simple ASCII file.

In this new paradigm, a set of programs and procedures on the UNIX side reads the input data file, determines the segments that are to be processed, ingests these segments (i.e., creates a separate file with only data to be processed as a unit), and runs a series of procedures that associated the navigation and calibration with the ingested file.

The processing recipes were also modified to copy the ingested files to the processing computer (instead of the single input file), and skip the sectorization (navigation/calibration addition).

Note that these changes were only in development, and had not yet been incorporated into the operational processing scheme. A comparison of the old and new calculation scheme follows.

Old	New
*PATHGAC_INIT	*NEWGAC_INIT
FTP the input pass file to the local gacinp directory	FTP the ingested pieces of the pass to the local gacing directory.
This job triggers a separate INGATCOR job for each piece of the pass.	This job triggers a separate SATCOR job for each piece of the pass.

==

Old	New
*PATHGAC_INGATCOR	*NEWGAC_ATCOR
This job ingests a piece, runs sector performs the atmospheric correction.	This job performs the atmospheric correction. It also stores the dataday information on that piece into a temporary pass/scan info file.
This individual job triggers a single SPACEBIN job to bin the single piece.	When ALL of the ATCOR jobs for a pass are complete, a single SPACEBIN job is run to bin all the pieces.

==

Old	New
*PATHGAC_SPACEBIN	*NEWGAC_SPACEBIN
This job bins one L2 piece into a 9km file	This job bins all pieces for the pass pst files, using the dataday information retrieved from the temp. scan file.
When ALL of the SPACEBIN jobs for a pass are complete, the ORBIT job is	The single SPACEBIN job triggers the ORBIT job.

triggered.

===

*PATHGAC_ORBIT	*NEWGAC_ORBIT
This job collects the SPACEBIN pieces, executes the ORBIT job ftp copies the results to the collection machine, deletes input files.	This job performs a ftp copy of the ORBIT file to the collection machine, deletes input files.

B.1.2.6 June Development

The processing scheme changes developed last month were implemented into the operational systems. Additional development occurred to accommodate the change in type of input file at the data time 92076 (NOAA L1b tape format to direction telemetry format). Files after this data were obtained directly by the RSMAS group, and the differences in file header information requires the use of a different scanner and ingester. Other development during the month centered on adjusting command files and programs to the new processing paradigm. While most problems have been cured, there are still some problems that occur in the scan/ingest section that must be resolved.

B.1.3.1 Other activity

Ran 1988 (Jan-NOV) using algorithm coefficients and model presented at Science Working Group meeting in March.
Processing rate, 7.5 days for 140 data days.
A new program path filler uses Laplacian relaxation to fill cloud areas. Use of Reynolds 1 deg, 1 week analysis as a quality control reference map does not provide a reasonable average reference value for areas with high gradients, e.g., South Atlantic Bight to Gulf Stream in Winter, Spring. Using filler program on good 3 week data to produce global filled 9km reference map, then run cloud mask program to check data validity, then produce new filled reference and run cloud mask.

Warner converted jukebox software to run on alpha

Vicki converted process control server to run on alpha
Expect that processing rates could double with faster archive access and network delivery, faster process control server response could improved present 15-20 data days/day to 30-40 data days/day.
Vicki processed 1987 and 1988, generated global products at 1 deg, 36, 18 and 9 km, distributed

Using matchup data base and radiative transfer modeling to examine limitations in present SST retrieval equations

Matchup database testing for N-11 from Nov. 88 until June 91.
Found change in sensor coinciding with Pinatubo eruption, effects global data by ~0.3C.
Started trial N-11 processing to examine transition from N-9 to N-11.
Due to difference in nodal time, approximately 1 orbit checking buoy record to determine amount of heating due to time difference, ~0.25c

B.2 Client/Server Status (S)

B.2.1 Client/Server Development

The following is a list of accomplishments for the 1st half of 1995 concerning Client/Server processing:

1. The ingester output file name assignment was been modified. In the past, when given a start line and an end line, the ingester output slightly different file names from those the input control specified. As modified the ingester will force a file name by passing the name to ingester through an environment variable. Although workable, this modification was not generally acceptable; the current solution is to run the ingester as usual and then acquire the new file name from the output directory and append the info the
the scan-info file.

2. A scan program for scripp format on Modis, named scripp_geninp was completed.

3. The program that computes the dataday starting and ending strings is finished and tested; the following provides some test results:

```
    file_stem    aday  dday   asc_beg      asc_end      DSC_BEG      DSC_END
91001001234  91000 91001 90365025126 91001024237 90365141022 91001140
91001012234  91000 91001 90365025126 91001024237 90365141022 91001140
```

4. A self contained program, pathgac_spacebin.dsp, to compute the data day time strings has been created; it has been made into a mice program so that time strings can be accessed from inside the dsp command procedure.

5. The first cut of the AVHRR Processing documentation has been completed and is being circulated internally for comment. The documents on the web is roughly presentable, although incomplete, and contains an explanation of data handling. This documentation will be linked to the algorithms (from Matchup documents) and to scheduling (from autoproc documents).

6. After cleaning up the code, the scan programs have been checked in for release control. Included in the changes are different satellite types.

7. Conversion of the program to Fortran90 has been discussed; this extension will be pursued after the creation of a GUI for the ap program.

B.3 Matchup Database <u>(P)</u>

During the first half of the year we completed version 18 of the AVHRR Pathfinder Oceans Matchup Database for the period 1985-1993. The database was delivered to JPL for public distribution. A document was prepared describing the compilation of the matchup database and its contents. This version was circulated for comments to L. Smith (Old Dominion University), P. Cornillon (URI) and J. Vazquez (JPL). The document was reformatted as an HTML file which can be accessed through the World Wide Web. JPL is in the process of adding the appropriate links to their Pathfinder WWW page.

Efforts to expand the temporal and spatial coverage of the matchups continued with the acquisition of a new data set including observations collected by moored buoy in the northeast Atlantic. These data were acquired from the UK Met Office. We also started the collection and reformatting of data for 1994. With regards to data prior for the period 1981-1984 (NOAA-7), all the in situ data were formatted and extraction lists were generated. The extraction of AVHRR data for those years will begin as soon as the GAC data are obtained from NASA-GSFC. With regards to 1981 in situ observations, drifting buoys from MEDS were missing in the available data set for July-December 1981, which we had originally obtained from NASA-GSFC. The missing drifter data were ordered directly from MEDS and added to the existing databases.

B.4 DSP Support (M)

B.4.1 Testing:
None listed

B.4.2 Modifications/Additions to DSP:
TIRPACK: New ingester to convert level-1b disk files ftp'd from NOAA to DSP header disk files that can be ingested using tiros.
Removed unused variables from all source files for SGI Power (64-bit) machines.
Use only single quoted strings in format statements.
Change Hollerith strings to single or double quoted strings per context.
IMGFILE parameter must be of correct type, can't use explicit constant.
LOADCOL: New program to read HIST line average files and create a dsp imag file (ala loadnohed) with one column per line average.

B.4.3 Problems fixed:

SEAWiFS support:

SMAP9-HDF:
Put remap parameters in command line.
Pigment, chlor, and k49 bands were mixed up.
Use WIFSROOT definition from make.prog.
Change the default for binning algorithm to seawifs style.
Fix different versions of flag handling.
Add third argument to bin9kminit.

Update for use in smap9-dsp to output dsp images instead of hdf l3m files.
Add binning model type to the associated data block.
Use only single quoted strings in format statements.
Take xspace and yspace out of command line since they aren't used for pst files.

SSBIN-HDF:
Use gsfc v4.2 i/o routines, add new temp/satz test,
fix binning to use left and right edge of bin instead of centers.
Use WIFSROOT definition from make.prog.
Change the default for binning algorithm to seawifs style.
Fix different versions of flag handling.
Add third parameter to bin9kminit.
Add binning model type to the associated data block and the command line;
add extension to input file name in "infiles".
Use only single quoted strings in format statements.

STBIN-HDF:
Use tilt; both gsfc and miami quality determination;
changes for gsfc v4.1 i/o routines.
Use gsfc v4.2 i/o routines.
Use WIFSROOT definition from make.prog.
Change the default for binning algorithm to seawifs style.
Fix different versions of flag handling.

ANLY8D:
Add optimization for epsilon-model choice across scan.
Start modifications for parameter file inputs.
Parameter file input complete.
(Needs to be rationalized against required input parameters.)
Remove unnecessary files.
Add documentation on parameter inputs.
Add checks to file opens to preclude invalid file names.
Increase maximum size of full filenames to 128 characters.
Update documentation for new arguments and parameter file input.
Fix typo in generation of La670 and La865.
Always treat 670 as aerosol band.
Add calhdf parameter for sensor calibration file.
Add method to specify sensor calibration filename from command line.
Print out selected sensor calibration filename.

Add diagnostics for epsilon carry-over algorithm function.
Change command-line syntax to allow 'SUNGLINT1=0.005'.
Move the make.prog include up so that WIFSROOT is defined.
Use $DSPROOT instead of /usr/dsp.
Return both La670/Lw670 to caller.
Change loops to only calculate items that will be used later (mostly
cleanup).
Tau calculation uses reflectance instead of radiance.
Fix makefile use of DSPLIB.
Rearrange cocco-algorithm coefficents to match preferred equation
form.
Change LwXXX to nLwXXX to match cocco-algorithm.
Another change in parameter order and default values for cocco-
algorithm.
Add third parameter to bin9kminit.
Add include file for bin9kminit model type.
Use 9km not isccp model.
Add CVS headers to some files.
Update copyright notices.
Don't pass explicit 0, put in variable of correct type and pass that
instead.
Add routines to parse input parameter MSKFLG into equivalent
output bit values.
Update to support wang2.f interface changes (flags2_pc).
Explicitly time execution of each of the four tests.
Pass in 'in_files' output value to L2 open.
Add MSKFLG parameter.
Rename NMCx and TOVSx parameters to METx and OZONEx
(respectively).
Add additional diagnostic flags to epsilon carry-over optimization.
Decode MSKFLG input string (though not used in subsequent tests).
Rename NMCx and TOVSx to METx and OZONEx.
Construct in_files and proc_con documentation strings
for L2 file.
Update output text for additional (flags2_pc) diagnostic flags.
Add new routine - splitandappend.rat for filename processing.
Implement proc_log output variable.
Change usage of in_files output variable.
Increase precision of intermediate variables in rho_a_sub_quad
to stabilize epsilon calculation (problem found by G. Fu).
Force delta phi input value into canonical range of [-180,+180].
Remove unused arguments from call to get_l1a_openf (trailing 5
variables).

ANLY,ANLY2D,ANLY6D,ANLY7D,ANLY8D: Change image file variables type from int to IMGFILE.
COLORSHR7:
Add routine to allow passing of climatology file names from command line.
Add additional climatology file entries.
Correct typo in handling of TOVS2 file.
Rearrange code to simplify all file handling.
Print out names of climatology files.
Pointer must be valid before string can be checked.
Image file variables are type IMGFILE, not type int.
Only build ephs/reflec/raylei/sunang2 once (they are in SPHLIB).
Limit status values from get_ancillary to 4 bits in the resulting
    composite status returned by get_climatology.
COLORSHR: Image file variables are type IMGFILE, not type int.
    Only build ephs/reflec/raylei/sunang2 once (they are in SPHLIB).
COLORSHR8: Image file variables are type IMGFILE, not type int.
    Only build ephs/reflec/raylei/sunang2 once (they are in SPHLIB).
    Force delta phi (angle) into canonical range of [-PI,+PI].
COLORSHR5: Only build ephs/reflec/raylei/sunang2 once (they are in SPHLIB).
LIB/DISPLYSHR/GETCMD.C: Exit with last reply status if last 'command' fails.
LIB/HDF/DFGR.C: Fix syntax error in if statement. Had '=' (assignment)

Pathfinder/MODIS support:

PATHNLC:
Use a coefficient file for all satellite/date combinations.
Add new temp/ satz test. Use <=0.7 and <=1.8 instead of <.
Change allb from logical to 0 for just sst, 1 for all bands, 2 for some bands (bright4, bright5, ch4m5, mask1, mask2, sst, hemsst, mnlsst); change some debugs.
Use different coef files.
Use only single quoted strings in format statements.
Change Hollerith strings to single or double quoted strings per context.
CONVRT: Fix string handling.
PATHTIME: Add comments.
    Add binning model type to the associated data block.
    Use only single quoted strings in format statements.
QRMPACK: Exit with good status.

SCRIPP: On unix, don't exit if yz0 is not defined.
Add line number to "check registration" debug.
Initialize variable.
Get year from header in some Sea Space files.
LIB/IO/AUTOOPEN.C,GET.C,PARSER.C:
Add support for parfile= keyword
LIB/IO/GET.C: Clear all nodes before reparsing, not just the left-pointing ones.
LIB/IO/ASSOC.C: Fix incomplete comment.
LIB/IO/BADIMGCHK.RAT: Image file variables are type IMGFILE and not int.
LIB/IO/GET.C: Exit to shell with non-zero status if error status and no CALLER.
INGEST/LIB/READLEVEL1B.RAT: Add read routine for raw NOAA Level-Ib disk files.
Remove SCRIPP specific functionality. Correct conversion errors.
TIROS,TIROSSCAN,RLREAD,NMFS: Increase size of variables used with
satellite library.
PATHBIN: Add ISSCP binning algorithm.
Add third argument to bin9kminit (model type: MDL_9KM [1] or
MDL_ISSCP [2]).
First argument to bin9kminit is tiny bin resolution (use 16 for 9km) -- must be a power to 2 to be MODIS compliant, routine will accept any positive value.
Update test program to completely test both the 9km and ISSCP routines.
Use 'make test9km' to build test program.
Add third parameter to bin9kminit.
Add more complete test program (try9km.rat) for bin9kmf.rat.
Test both 9km and ISSCP modes for all routines.
Add additional argument range checks to bin9kminit.
Add CVS headers to some files.
Update/add copyright headers.
Add binning model type to the associated data block and command line; change allb from logical to 0 for just sst, 1 for all bands, 2 for some bands (bright4, bright5, ch4m5, mask1, mask2, sst, hemsst, and mnlsst).
Increase AABINS to allow either Miami or ISCCP algorithms at 9km.
Correct name to ISCCP from ISSCP.
Put constant in variable of correct type instead of passing a constant.
Use only single quoted strings in format statements.

Use float(ii) just like float(jj).
PST2OA: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
PATHSPC: Add third parameter to bin9kminit.
Add binning model type to the associated data block and command line.
Use only single quoted strings in format statements.
Fix to handle up to 6 bands to sum (instead of just sst)
PATHMOS: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
PATHMAP: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
Use only single quoted strings in format statements.
PATHFLT: Add third parameter to bin9kminit.
Use not(###) instead of !### inside and(x,y).
Add binning model type to the associated data block.
PATHFILL: Add third parameter to bin9kminit.
Change '!###' to 'not(###)'.
Add binning model type to the associated data block; allow for some bands (more than just sst) in input file.
OA2PST: Add third parameter to bin9kminit.
Add binning model type to the associated data block and command line.
Use only single quoted strings in format statements.
IMGFILE parameter must be of correct type, can't use explicit constant.
Fix binning to use left and right edge of bin instead of centers.
MOSAIC9: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
Type of image file pointer should be IMGFILE and not int.
Change image file variables type from int to IMGFILE.
IMG2PST: Add third parameter to bin9kminit.
Add binning model type to the associated data block and command line.
Use only single quoted strings in format statements.
Fix binning to use left and right edge of bin instead of centers.
IMG2BIT: Add third parameter to bin9kminit.
Add binning model type to the associated data block and command line.
Put constant in variable of correct type instead of passing a constant.
Use only single quoted strings in format statements.
GSFCBIN9: Add third parameter to bin9kminit.

Add binning model type to the associated data block and command line.
Use only single quoted strings in format statements.
CZCSMAP9: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
CSBIN: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
Use only single quoted strings in format statements.
CMOS: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
Put constant in variable of correct type instead of passing a constant.
Change image file variables type from int to IMGFILE.
CMAP9: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
BIT2OA: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
BIT2IMG: Add third parameter to bin9kminit.
Add binning model type to the associated data block.
Put constant in variable of correct type instead of passing a constant.
MAKE-BSD: Add target for 'Power' series SGI machines.
Use target for 'Power' series SGI machines.
Add C compile option to allow 64-bit build on SGI 'Power' series.
Fix incomplete comment consumed code to remove leading blanks.
Consistently use $(MAKE) instead of 'make'.
Remove files not used to build a localized version of 'pmake'.
AVHRRSHR5: Change image file variables type from int to IMGFILE.
Use only single quoted strings in format statements.
Only build ephs/reflec/raylei/sunang2 once (they are in SPHLIB).
VHRR: Change 'call GETADR(x,y)' to 'x = IADDR(y)'. More portable.
Change 'call GETADR(x,y)' to 'x = IADDR(y)'. More portable.
Use only single quoted strings in format statements.
Remove duplicate (decodr) directory (lib is correct one to keep).
Localization change.
Allow 'old-style' ingest files to be accessed as 'read-only'.
RATFOR: Add missing return value to 'caslab'. Fake return value in 'emalloc'.
Add symbol 'SGI_64' for SGI Power (64-bit) machines.
Terminate a 'non-CHARACTER' text string with a NULL byte for SGI Power-series.
MICE: Correct syntax error (had GenC instead of generateC near line 233).

PUTNLPPT7: Use 'iostat=ios' instead of 'end=' to detect end of file on read.
XFBD: Add lines to SGI Power series (XtoX[hl] functions).
Copy plane sizing into return packet (_GetPlaneSize).
Correct usage of callbackList2.
Was setting it up and then using callbackList instead of callbackList2.
PATHCLOUD: Add binning model type to the associated data block.
PATHCOMP: Add binning model type to the associated data block.
Use only single quoted strings in format statements.
Fix to handle any number of input bands, instead of just sst.
PATHMASK: Add binning model type to the associated data block.
PATHREF: Add binning model type to the associated data block.
TQ2023: Move to correct directory.
TQ2024: Move to correct directory.

TROUTC,WMEAN,STATS,SHRINK,PIXRD,PATHDR,KEY8,COMPARE,JHIST, 9KLM2IMG,HIST:
Put constant in variable of correct type instead of passing a constant.
WMEAN: Use only single quoted strings in format statements.
SMOS: Change image file variables type from int to IMGFILE.
TINYBIN: Change image file variables type from int to IMGFILE.
DMPCNT: Set FORTRAN LUN value non-zero.
GETCOM: Put constant in variable of correct type for image file value.
Text in format statements should be 'text' and not "text".
Rename 'access' to 'accessfile' to avoid RTL conflict.
instead of '==' (comparison).
DMPHDR: Change Hollerith strings to single or double quoted strings per
context.
MERGEG: Change IMAGE0 and GRAPHIC0 to generic 'IMAGE0' and 'GRAPHIC0' in
error msgs.
NEWIMAGE: Modify lat/lon bounds code to work (better) for Robinson projection.
Change adjustment to scaled real value in case output line is not 1024 pixels.
LIB/VMSFORLIB: Don't build getadr.c or idate.c.
Rename access and unlink to not conflict with unix RTLs.
SHPSPH: Rename 'unlink' to 'unlinkfile' to remove RTL conflicts.
Rename fnmin1/fnmax1 to fnmin2/fnmax2 to remove conflict with SPHLIB.

MCSST: Don't include ephs/reflec/raylei/sunang2 if already available in SPHLIB.
AES: Input statement is requesting two input values.
  Upgrade missing line handling code to latest method.
  Insert held line after missing line gap.
  Some little fixes for vms/unix differences.
AES10: Input statement is requesting two input values.
  Separate out modules used only by VMS.
  Upgrade missing line handling code to latest method.
  Insert held line after missing line gap.
  Some little fixes for vms/unix differences.
MACE2: Check for invalid image start date.
TRAVEC: Correct decode/format statements to work on PMV input file.
  Add some conditional debug statements.
  Add error output with line counting to locate bad input records.

B.5 Direct Project Support

MODIS, Pathfinder- Interface with Navy to explore near real time access to *in situ* data sources: Our group met with G. Mason and D. May representing NAVOCEANO to define potential options to access existing data sets and examine possible extensions to these data sets. In turn our group will provide NAVOCEANO with present Pathfinder SST programs. Together our groups will define and implement procedures to generate equation coefficients for real time SST product generation and quality control.

MODIS ocean team members have been asked to submit updated algorithms and input/output requirements. The updated algorithms will be included in the programs submitted for the next Beta release. The input and output requirements will be used to define the product files.

Gridding, Data Day Issues - Discussions concerning selection of appropriate Level 3 grids were undertaken at the Spring MODIS meeting and continued at the Sante Fe IWG. Our group and the CERES team are conducting a series of tests to determine effects of the grid definition when data at one resolution is converted to another resolution. We are studying the problem by directly generating fields of SST (both cloudy and valid SST values) at low and high resolution. The high resolution data is then combined to form a field at the lower resolution and the two low resolution fields are compared after each is mapped to a common map projection.

Initial discussions with the CERES team suggest that selection of a “data day” definition also will be a contentious issue. Selection of a “data day” should reflect the individual product requirements. A single definition for EOS would not reflect current practice or geophysical reality.

initial beta delivery- SeaWiFS programs for oceancolor products, SeaWiFS programs use SeaWiFS HDF I/O for L1, ancillary fields, and output products.
meeting with ocean support group of SDST in Miami
define needs for input (L1b, cloud, land/water boundary, ancillary fields [surface wind speed, u,v; atmospheric pressure, relative humidity, ozone], geometry, latitude, longitude, satellite and solar zenith angle, azimuth angles), access to PCF, HDF output files for products and quality fields.
Pathfinder SST program requires HDF I/O

Pathfinder SWG meeting to discuss post Mt. Pinatubo processing. Processing prior to the eruption used algorithm coefficients that spanned a several year time frame and included a linear correction in time to partially account for long term drift. The rapid changes in absorbing aerosols following the eruption introduced errors that exceeded those seen in prior years. A procedural change was introduced where equation coefficients are recomputed on a monthly basis. The approach reduced both long term and seasonal errors.

The present NOAA NLSST equation used in the Pathfinder SST processing utilizes the blended Reynolds analysis to provide the reference SST. Dick Reynolds (NOAA) was asked and has delivered maps for a one year period showing the location of the *in situ* data that is incorporated in his analysis. The location data is being used together with an analysis of the magnitude of the magnitude of the Reynolds-Pathfinder difference to help understand the relative behavior of the fields. The fields tend to agree well where a significant number of *in situ* points have been included in the Reynolds analysis. The fields differ the most when the Reynolds analysis relies substantially on the NOAA satellite SST.

The following hardware has been acquired to support MODIS processing

Disk Expansions - 22 4 GB Barracuda - 88 GB
24 8 GB Elite 9 - 192 GB
280 GB
1 PCI RAID controller for 2100 Server
2 ISA RAID controllers for 2100 Server

Tape Expansion - 1 DEC 5 TB Tape Store

Computer Expansion - 2 DEC 2100 Servers [4 275 Mhz processors, 512 MB memory each]

Network Expansion - FDDI Expansion -
1 DEC Concentrator 500 Additional 8 port Card
5 DEC PCI FDDI Adapters for workstations and servers
4 DEC Turbochannel FDDI Adapters for workstations
ATM Expansion -
4 DEC quad line cards for GigaSwitch/ATM
4 DEC Turbochannel ATM Adapters for workstations
2 DEC PCI ATM Adapters for workstations
ISDN Expansion
4 Etherner/ISDN Adapters

The following software has been acquired to support MODIS processing

DEC OSF1 V3.2 has been installed on most processing platforms.
DEC OSF1 V3.4 [beta] has been installed on some processing platforms.

The following support has been acquired to support MODIS processing

Sybase support has been renewed.
TGV support has been renewed.
DEC hardware maintenance has been renewed.

Expected:

Computer Expansion - 5 DEC Alphastations 600 5/266 [266 Mhz processor, 128 MB memory each] will arrive in Q3.

DEC OSF1 V4 [beta] will arrive in Q3.

C. FUTURE ACTIVITIES

Future activity will focus on evolution of the initial Beta software delivery from a SeaWiFS algorithm suite supported by SeaWiFS I/O routines to use of prototype MODIS algorithms using the MODIS API as these routines become available.

We will receive beta versions of new operating systems for both SGI (the 64 bit system) and DEC, FORTRAN 90 compilers for both systems, and enhanced ATM communications software and hardware. Results form these tests will communicate to the SDST.

We will explore the possibility of using elements of the HUGHES/EOS process control environment in Miami to implement an EOS compliant processing framework. Discussions concerning this approach have been initiated.